REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	17 MAY 1996	PROFESSIONAL PAPER

4. TITLE AND SUBTITLE

IMPROVING MUNITION SIMULATION FIDELITY THROUGH USE OF AN ORDNANCE SERVER

5. FUNDING NUMBERS

6. AUTHOR(S)

JOHN DICOLA, DAVID MUTSCHLER,
LAWRENCE ULLOM, PETER FISCHER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)

COMMANDER
NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION
22541 MILLSTONE ROAD
PATUXENT RIVER, MARYLAND 20670-5304

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

COMMANDER
NAVAL AIR SYSTEMS COMMAND
1421 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VA 22243

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A recurring problem in Distributed Interactive Simulation (DIS) is the use of different munition models by different entities in the same exercise. Even when these entities simulate the same weapon, the use of different munition models with varying levels of fidelity can unjustly favor the performance of one combat platform over another. since validated weapon simulations with realistic fly-outs and trajectories may perform differently than less complicated models, the accuracy of the simulation exercises can suffer greatly. This paper describes a solution wherein munition simulations are separated from launching platform simulations and incorporated into an ordnance server. This server provides multiple entities in an exercise with an uniform set of munition models. When a munition is launched, the server utilizes the best available model to ensure the most correct simulation of he munition and controls it appropriately. Thereby, overall simulation fidelity is improved. In addition, the ordnance server facilitates simulation design and management since munition models may be developed, tested, and verified independently of the simulation of any launch platform.

14. SUBJECT TERMS

Distributed Interactive Simulation (DIS);ordnance server; Manned Flight Simulator

15. NUMBER OF PAGES
20

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	N/A

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Encl (4)

IMPROVING MUNITION SIMULATION FIDELITY THROUGH USE OF AN ORDNANCE SERVER

John Dicola, David Mutschler, Lawrence Ullom
ACETEF/MFS, Naval Air Warfare Center -- Aircraft Division Patuxent River, MD

Peter Fischer
J.F. Taylor, Inc.

Abstract

A recurring problem in Distributed Interactive Simulation (DIS) is the use of different munition models by different entities in the same exercise. Even when these entities simulate the same weapon, the use of different munition models with varying levels of fidelity can unjustly favor the performance of one combat platform over another. Since validated weapon simulations with realistic fly-outs and trajectories may perform differently than less complicated models, the accuracy of the simulation exercises can suffer greatly. This paper describes a solution wherein munition simulations are separated from launching platform simulations and incorporated into an **ordnance server**. This server provides multiple entities in an exercise with a uniform set of munition models. When a munition is launched, the server utilizes the best available model to ensure the most correct simulation of the munition and controls it appropriately. Thereby, overall simulation fidelity is improved. In addition, the ordnance server facilitates simulation design and management since munition models may be developed, tested, and verified independently of the simulation of any launch platform.

Introduction

Distributed Interactive Simulation (DIS) was designed from the beginning to bring a wide variety of different simulators and simulations together in a unified synthetic world. This primary goal leads to an inherent problem with the use of different munitions by different entities in the same exercise. The use of different munition models with varying levels of fidelity can unjustly favor the performance of one combat platform over another. This raises the issue of a "fair fight" when using data from these exercises to make acquisition decisions, evaluate measures of effectiveness in training, or develop strategic doctrine.

Due to the different requirements of each of these groups, it is not logistically possible to pick a standard model and distribute it for inclusion in each simulator in an exercise. This paper describes a solution wherein munition simulations are separated from launching platform simulations and incorporated into a server. The server takes over the modeling of weapons in an exercise from the initial fire event generated by the launching entity and controls the munition through its termination. This approach can increase the overall fidelity of an exercise. It also aids configuration management by separating the weapon models from the launch vehicle.

An ordnance server can provide a level playing field by incorporating models of equal fidelity for all weapons used in an exercise. Munition modeling is no longer driven by the requirements of particular simulators. Moreover, the exact level of fidelity can be driven by the particular exercise requirements.

An implementation of this approach, entitled the Ordnance Server (OS), was developed at the Manned Flight Simulator (MFS) laboratory of the Air Combat Environment Test and Evaluation Facility (ACETEF) at NAWC-AD Patuxent River, MD*. It was originally used with man-in-the-loop flight simulators to support classified and unclassified munition modeling. The method was later tested with synthetic entities in a large scale training exercise and proved viable. Additional research is needed to determine efficient mechanisms for passing prelaunch and inflight data from the launch platform to the munition model. With continued development, the Ordnance Server can further improve the validity of results obtained from distributed simulation exercises.

---

* Any correspondence can be sent to Commander, Attn.: Alexandra Wachter, Code 5.1.6.1, MS: 3. Naval Air Warfare Center -- Aircraft Division, 48140 Standley Road, Patuxent River MD 20670-5304 or via e-mail at wachteram%am6@mr.nawcad.navy.mil.

Standard DIS Missile Simulation

In the DIS protocol, defined messages known as Protocol Data Units (PDUs) are used to transfer information between simulators. Three types of PDUs describe a tracked munition simulation in a DIS exercise: Fire, Entity State, and Detonation. The Fire PDU is issued by the firing entity as an initiation of the munition. It contains information including the identity of the firing entity, the identity of the intended target, the type of munition fired, the initial velocity of the munition, and the location from which the munition was launched. Thereafter, Entity State PDUs are issued to describe the resulting trajectory of the munition. A Detonation PDU is issued at the end of the munition flight. It communicates the location of the burst (if there was one), the detonation result, and other related information. It is the target simulation's responsibility to assess the damage to the target from the information provided in the Detonation PDU.

Missile Simulation by Ordnance Server

Normally, all three types of PDUs are issued by the firing entity's simulation application. However, when the Ordnance Server (OS) is used, the firing entity's simulation application only issues the Fire PDU. The server is responsible for using the information from the Fire PDU, and other ground truth information available over the DIS network, to properly issue the Entity State and Detonation PDUs for that munition.

This methodology requires the same amount and type of PDU traffic as that required for a standard entity simulated munition launch. The network sees no additional load due to the utilization of the server.

Determination of Pertinent Fire PDUs

Before the exercise begins, the Ordnance Server (OS) is configured to simulate missiles for some or all entities from a particular site or application. Upon receiving a Fire PDU from one of these entities, the Ordnance Server determines if the type of missile fired is one that it has been configured to simulate. If so, the missile simulation is initiated. Otherwise, the Fire PDU is ignored. This configurable filtering gives the user a great deal of flexibility as to the extent to which the Ordnance Server is used, versus the extent to which an application simulates its own munitions.

This flexibility can also be used to further distribute the processing load. One Ordnance Server can be configured to simulate all munitions from one application, while another Ordnance Server is configured to serve a different application. In a different case, one Ordnance Server could be configured to simulate all guided munitions, while another one simulates the ballistic munitions. This flexibility is important due to the highly variant nature of the scopes and requirements for different DIS exercises.

Basic Functionality of Ordnance Server

All configuration and control of the Ordnance Server (OS) can be accomplished, by the user, via the OS's Graphical User Interface (GUI). The OS was designed to be maximally configurable. Beyond the previously described configurable filter criteria, the OS's GUI also provides the following functions:

- Selection of the terrain database to use for ground impact checks,
- Mapping of the missile type input (via Fire PDU) to munition model used,
- Freezing and resuming the munition simulation(s),
- Specification of the guidance method and aerodynamic behavior of generic munition models,
- Specification of the radar emissions produced by actively guided munitions,
- Selection of the fuse and warhead types of the munitions,
- Selection of the type of runtime feedback to be provided to the user, and
- Provision of other detailed simulation information.

The OS's GUI provides configuration data to the OS Executive. In return, the OS Executive provides runtime feedback about munition and OS performance to the GUI for display. OS configuration data can be saved in and loaded from configuration files to ease initialization during subsequent exercises.

The OS includes a set of built-in generic munition models with configurable guidance methods and aerodynamic behavior. When a Fire PDU is received via the DIS Gateway, the OS's PDU Engine routes the pertinent information to the OS Executive. If the Fire PDU is from a client entity of the OS and it's munition type is mapped to a generic missile model, the OS Executive then initiates the proper built-in generic fly-out with the selected guidance model. The built-in fly-out model keeps track of the current state of the munition model by way of a munition database that it continuously updates and accesses during the munition's flight.

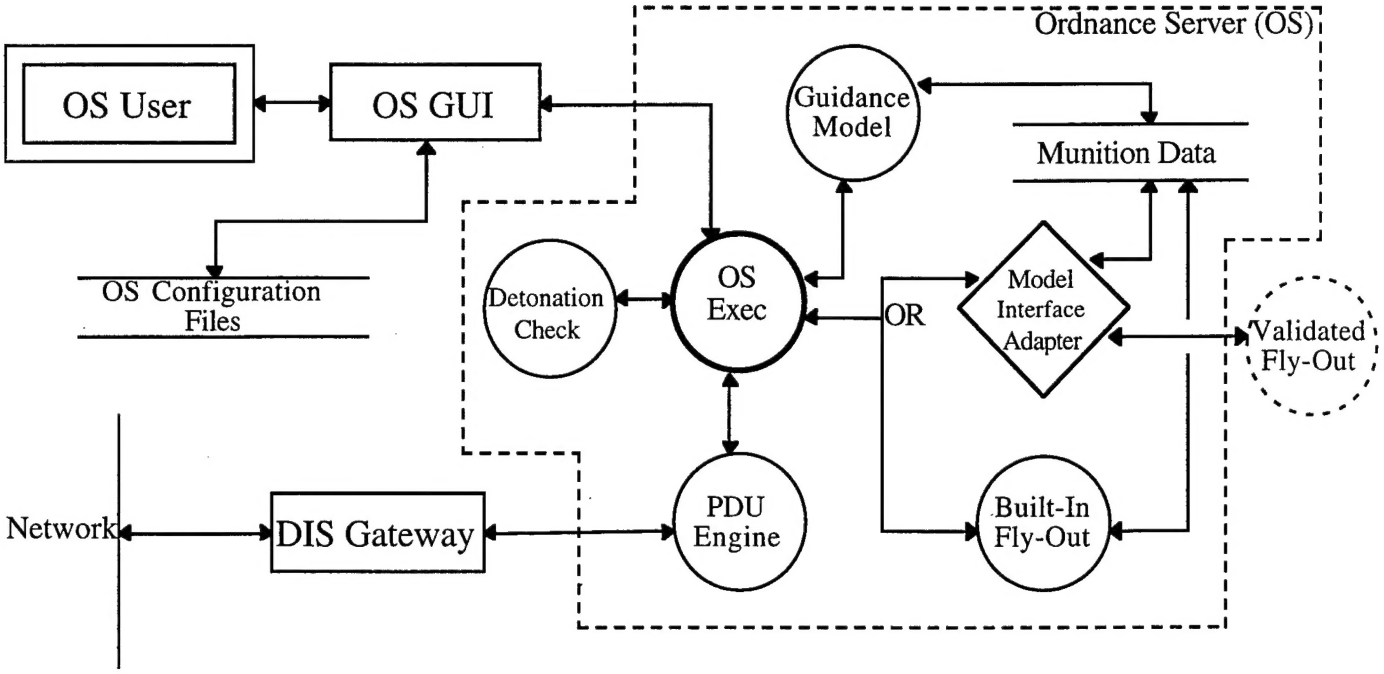


Figure 1: Simplified Block Diagram of Ordnance Server

The OS can simulate multiple concurrent munition fly-outs by simply adding new munition records to this munition database and then updating them in turn. Also during the munition flight, the built-in fly-out provides current data about the munition's position, velocity, and orientation to the OS Executive. The OS Executive routes this information to the PDU Engine for creation of the munition's Entity State PDUs. The PDUs are then broadcast over the network by the DIS Gateway.

During a guided munition's flight, the DIS Gateway keeps track of external entities and emissions that could affect the guidance of the munition. The OS Executive continuously provides this information to the guidance model so that it properly models the input to the missile sensors and guides the munition's fly-out model realistically.

Regularly, during all munition fly-outs, detonation checks are made. The criteria that cause a generic munition to detonate are highly dependent on the fuse type specified by the user at configuration time. The OS's generic munition fuse types can be altitude, depth, proximity, contact, or time fused. Detonations can be caused by external detonations, collisions, ground impacts, or entity impacts. Information about these external entities and events is provided by the DIS Gateway. When the munition's detonation criteria have been met, the OS Executive sends the detonation information to the PDU Engine for creation of a Detonation PDU. This PDU is then broadcast over the network by the DIS Gateway. The OS executive also deactivates the built-in fly-out model. Upon deactivation, the fly-out model removes the munition's data from the munition database so that no more Entity State PDUs will be issued for that munition.

Interface Between OS and Validated Munition Models

When a level of fidelity beyond that of the generic missile models is required, the Ordnance Server provides the use of validated munition models. These models are integrated into the Ordnance Server by developing an interface between the existing validated model and the existing OS code. This interface is called the Model Interface Adapter.

Functional flow within the OS during an interfaced external munition simulation is identical to that of a built-in fly-out model simulation (described earlier) with the following exception: the functionality of the built-in fly-out model is replaced by the union of the Model Interface Adapter and the interfaced validated fly-out model. The functions of the Model Interface Adapter are:

- To make the external munition simulation "look like" (as far as the type and quantity of data being

passed in and out) the OS's built-in fly-out model to the OS Executive, and

- To make the OS "look like" the simulation executive, under which the external model was designed to run, to the external fly-out model.

Most of the external munition models that the OS currently supports were originally developed for the Tactical Aircrew Combat Training System (TACTS). TACTS munition models have undergone a fairly extensive validation and verification process and are very good representations of the missiles they emulate. The TACTS missile simulations were also good candidates for integration into the OS because they are easily ported, have a well-defined interface, are capable of being called cyclically, and run in real time.

Some differences between TACTS models and the built-in OS models are location and orientation representations, required cycle periods, and the variety of runtime feedback produced. All of these differences were resolved by the Model Interface Adapter. For example, TACTS missiles provide extra runtime feedback as descriptive missile failure messages and probability of kill estimates. To ensure that this information is not wasted, additional functionality was added to the OS's GUI so that this data is provided to the user as needed. The probability of kill estimates are used to determine the detonation result reported in the munition's Detonation PDU.

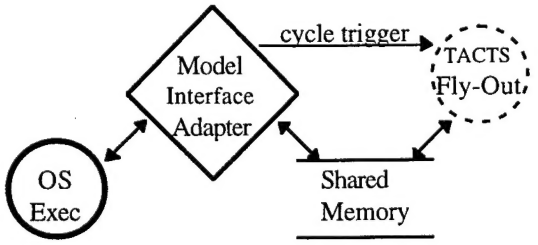


Figure 2: OS/TACTS Interface

The TACTS models were designed to interface with their simulation executive through shared memory blocks. These are accessed by the Model Interface Adapter by mapping equivalently sized data structures over the shared memory locations maintained by the TACTS models. The TACTS model's cycle time is controlled by the Model Interface Adapter. The adapter uses a TACTS routine to trigger the TACTS model and cause it to read and update the shared memory interface.

Available Munition Models

The TACTS munition models listed in the following table have been integrated into the OS. Additionally a Tomahawk Trajectory Generation Tool, developed by the Dahlgren Division of the Naval Surface Warfare Center, has been successfully interfaced into the OS. Additional TACTS and other types of validated munition simulations will be added to the OS to ensure that the OS can provide the best available model needed by prospective client applications.

TACTS Models	Common Name
AA-2B, 2D	Atoll
AA-8A, 8C	Aphid
AA-10A, 10B, 10C	Alamo
AA-11	Archer
AIM-7F, 7M, 7E2	Sparrow
AIM-9G, 9H, 9L, 9M, 9P3	Sidewinder
AIM-54A, 54C	Phoenix
AIM-120	AMRAAM
FIM-92	Stinger
R-550	Magic
SA-2	Guideline
SA-3	Goa
SA-4	Ganef
SA-5	Gammon
SA-6	Gainful
SA-7	Grail
SA-8	Gecko
SA-9	Gaskin
SA-16	
SA-N-3	Goblet

Table 1: TACTS Models
Currently Integrated into the Ordnance Server

Simulation Design and Management

Since the Ordnance Server runs independently of the simulation of any launch platform, munition models can be developed and integrated without having to account for internal differences between the various launching OS client applications. The programming language, structure, and ideology of the client applications are all completely transparent to the Ordnance Server. Any development of new or old munition simulations on the Ordnance Server automatically enhances the munition simulations of all client applications.

The structure and content of Fire PDUs is well defined by the DIS standard. Since the Fire PDU is used as the interface between the Ordnance Server and firing client applications, an Ordnance Server tester can assume that if a model performs correctly when launched by one application that issues properly encoded Fire PDUs, then it will work with all applications that issue properly encoded Fire PDUs. For instance, all testing of a Sidewinder launched from a DIS compliant F-18 simulation need not be duplicated for a Sidewinder launched from a separate, DIS compliant F-14 simulation.

Verification of the munition models is also facilitated by the use of an Ordnance Server. If munition models are modeled by the firing entity's simulation applications, the determination that a missile correctly approximates the actual behavior of it's real counterpart must be made for each application that integrates the missile model. Even then, there is no guarantee that duplicate missile types will behave identically from application to application. With the use of the Ordnance Server, verification needs to be completed only once for each of the types of munitions modeled. Since all simulations share the exact same set of munition models, consistent missile behavior is assured.

Conclusion

The Ordnance Server provides an efficient, logical means of separating munition simulations from simulated launching platforms. As a stand-alone munition simulator that initiates weapon launches via Fire PDUs, the OS provides a non-intrusive method for multiple simulations to share a uniform set of weapon models. It allows independently validated, high fidelity munition models to be easily integrated into a simulation or set of simulations. Sharing munition models leads to more realistic exercises and moves modeling and simulation closer to providing a "fair fight" by eliminating weapon fidelity variations.

Acknowledgments

The authors wish to thank the following people for their assistance in developing the Ordnance Server: Paul Maassel for concept exploration and refinement, Rob Kerr and Kimberly Neff for implementation, and Al Gramp and John Phillips for their support in integrating the TACTS models. Lastly, the authors would to thank Alexandra Wachter for organizing and leading this team and for providing the spark and support necessary for this paper's creation.

Authors :

Lawrence Ullom, John DiCola, David Mutschler

NAWCAD Patuxent River

Peter Fischer

J.F. Taylor Inc.

Information:

Alex Wachter; NAWCAD

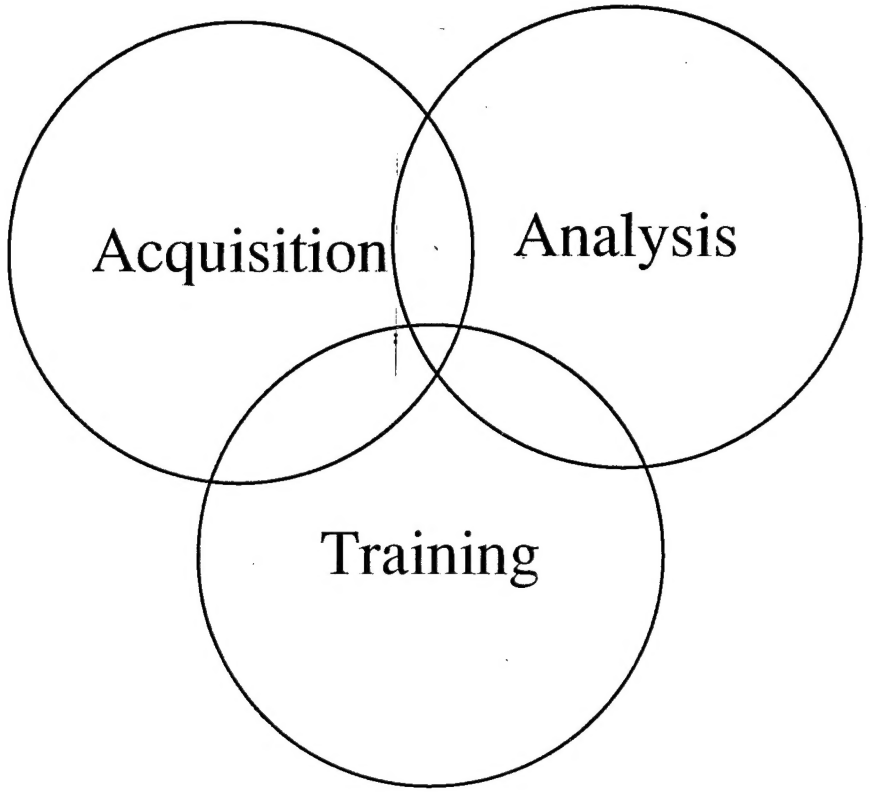
wachteram%am6@mr.nawcad.navy.mil

Introduction

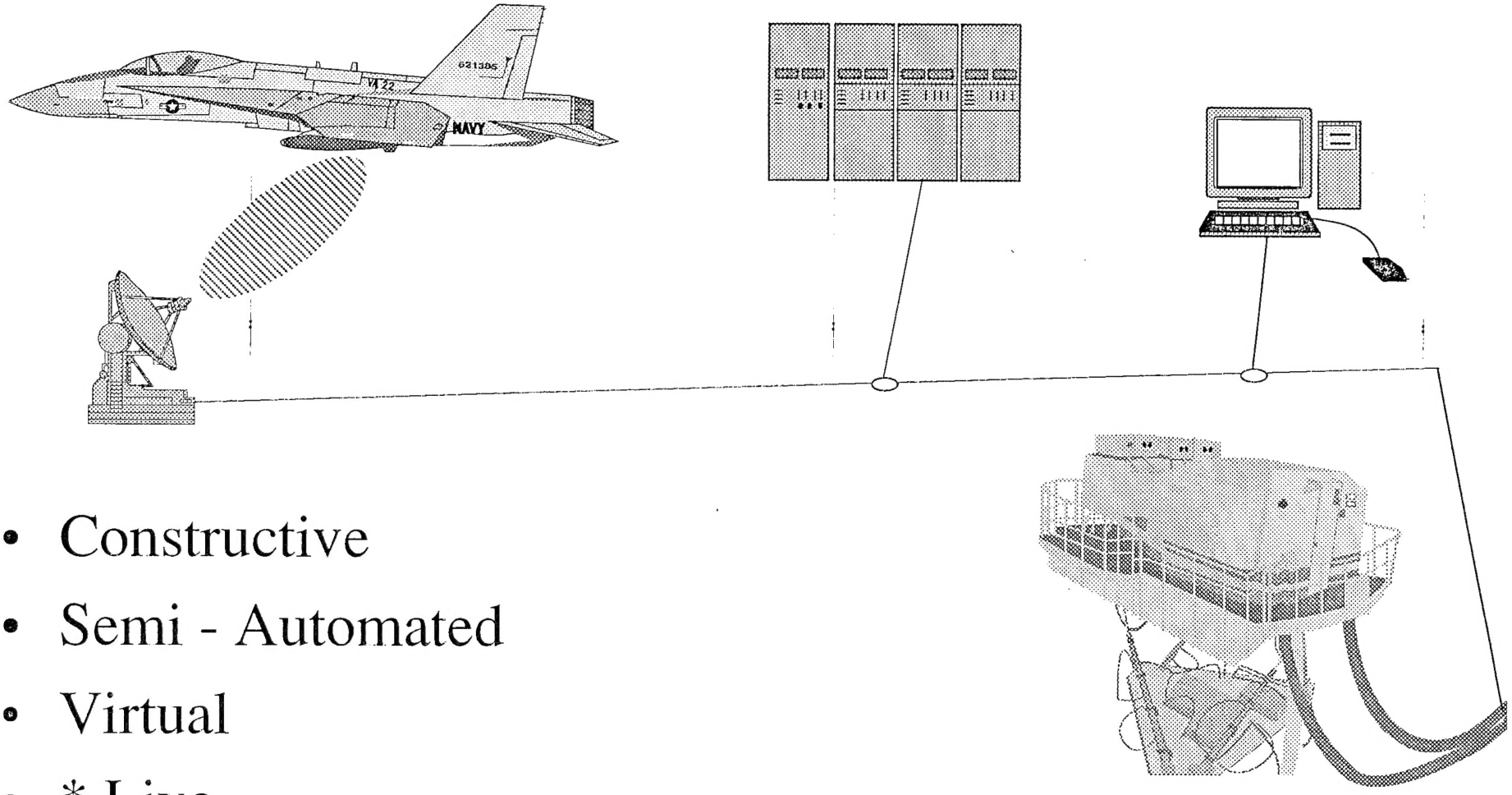
- Origins of Distributed Interactive Simulation (DIS)
- Guiding Principles
  - No central computer
  - Autonomous simulation applications
  - Standard protocol
  - Exchange of entity state data
  - Event and entity perception determined by each host
  - Dead reckoning to reduce bandwidth

Application Domains

- Acquisition
  - Simulate Before Build
- Training
  - Improved Joint Operations
- Analysis
  - Doctrine Development



Simulator/Simulation Systems Used



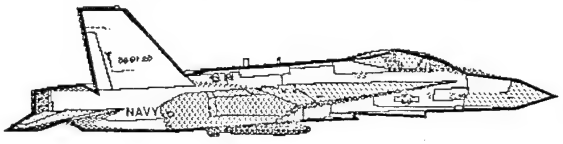
- Constructive
- Semi - Automated
- Virtual
- * Live

Inherent Problems

- Different Model Fidelity / Level of Detail
  - Differences in Capability
    - Uneven Playing Field
      - NOT a "Fair Fight"

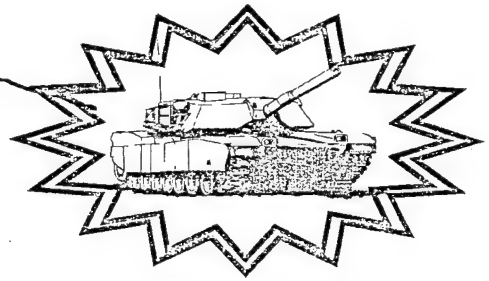
The DIS Weapon Simulation

Application simulation



- Launching entity launches weapon
- Launching entity sends a Fire PDU
- Launching entity sends ESPDUs of munition flyout
- Munition model detonates
- Launching entity translates detonation result
- Launching entity broadcasts Detonation PDU

Application simulation



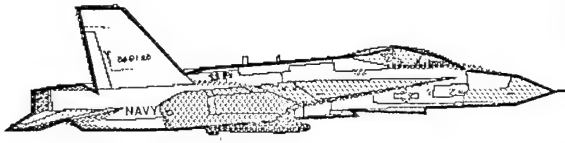
- Target entity receives Detonation PDU
- Target entity performs battle damage assessment

Weapon Simulation With An Ordnance Server

Application simulation

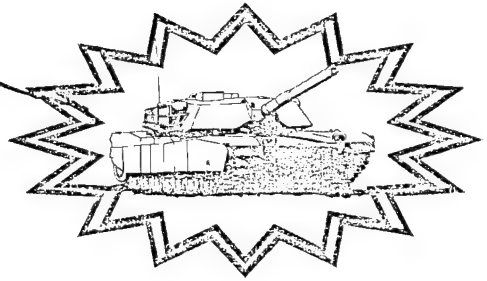
Ordnance Server

Application simulation



•Launching entity launches weapon
•Launching entity sends a Fire PDU
•

•OS reads Fire PDU off net
•OS performs munition flyout
•Munition model detonates
•OS translates detonation result
•OS broadcasts Detonation PDU



•Launching entity receives Detonation PDU

•Target entity receives Detonation PDU
•Target entity performs battle damage assessment

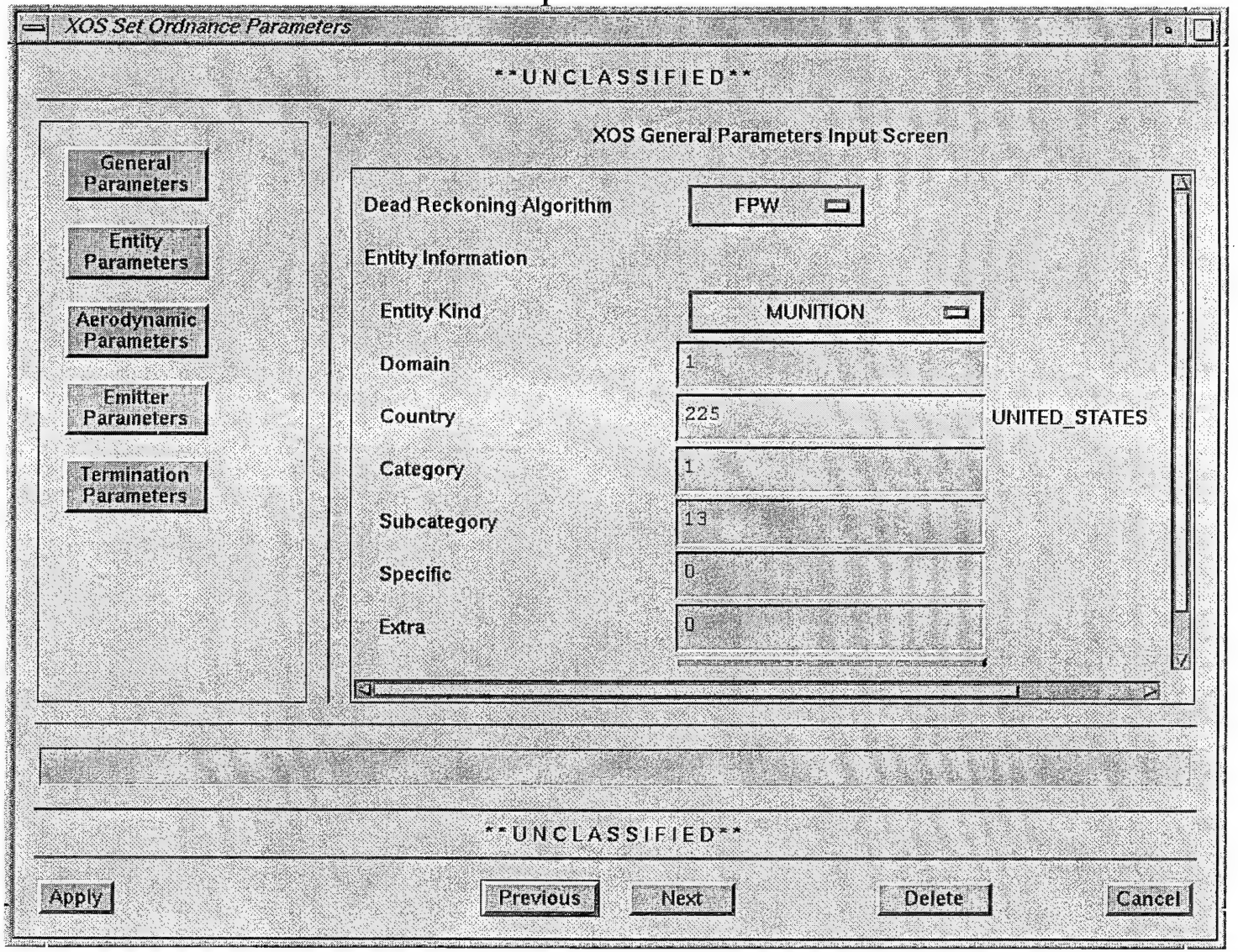
Pertinent Fire PDU Determination

- Pre StartX OS Configured With
  - "Parent" Site | Application | Entity
  - Entity Types for "Know" Weapons
  - Model to Use for Entity Type
- Post StartX OS Applies Filters to Fire PDU Based on Configuration Data
- Incomplete Fire PDUs Discarded

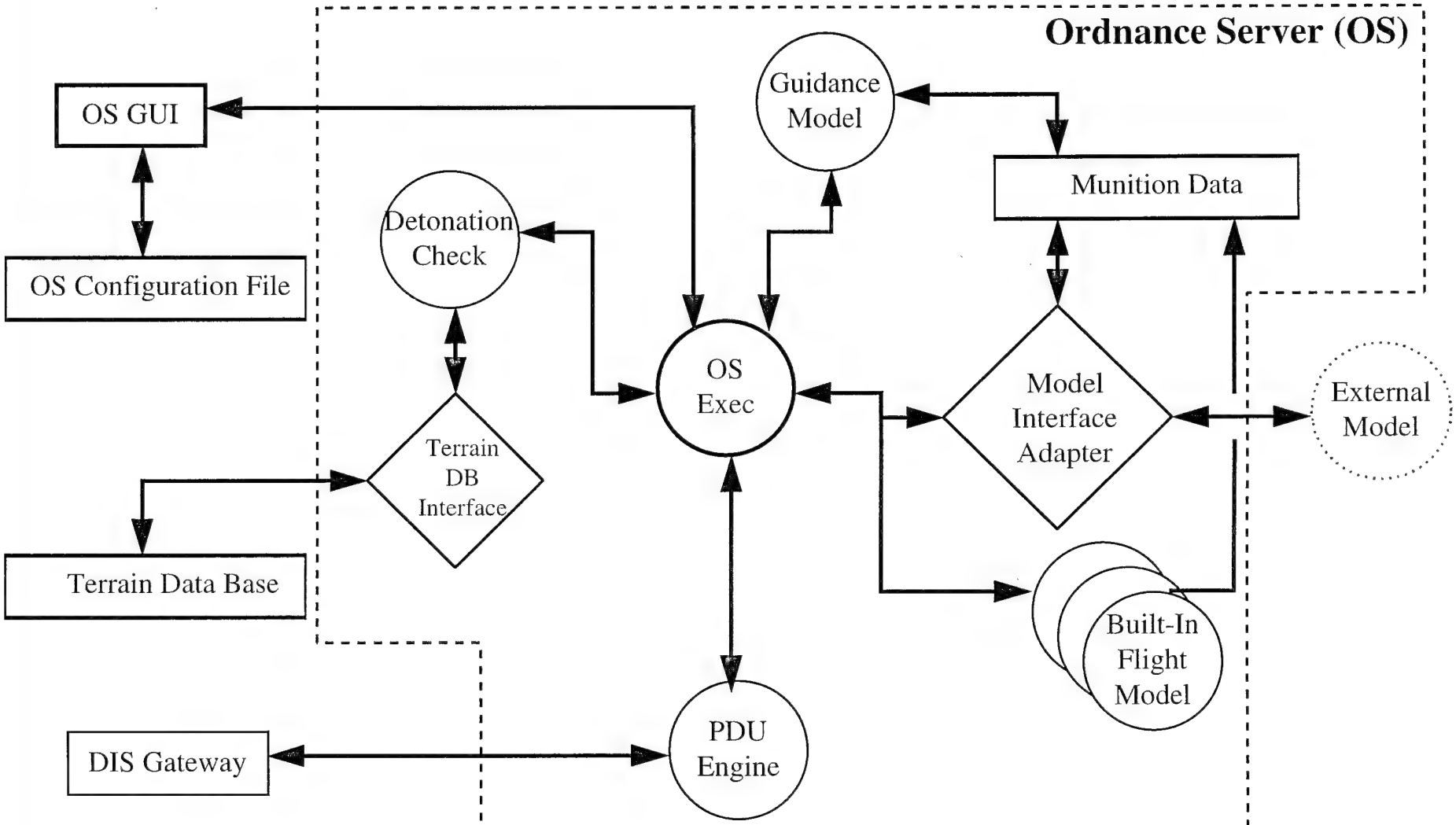
Ordnance Server - Graphical Interface

- Select Terrain Database
- Map Entity Type (from Fire PDU) to Munition Model
- Freeze/Resume Munition Simulation(s)
- Configure Guidance Method and Aerodynamic Parameters of Generic Model(s)
- Describe Radar Emissions Produced by Active Guidance Models
- Select Fuse and Warhead Types of Munitions
- Enable/Disable Types of Runtime Feedback
- Set Munition Specific Parameters for External Models

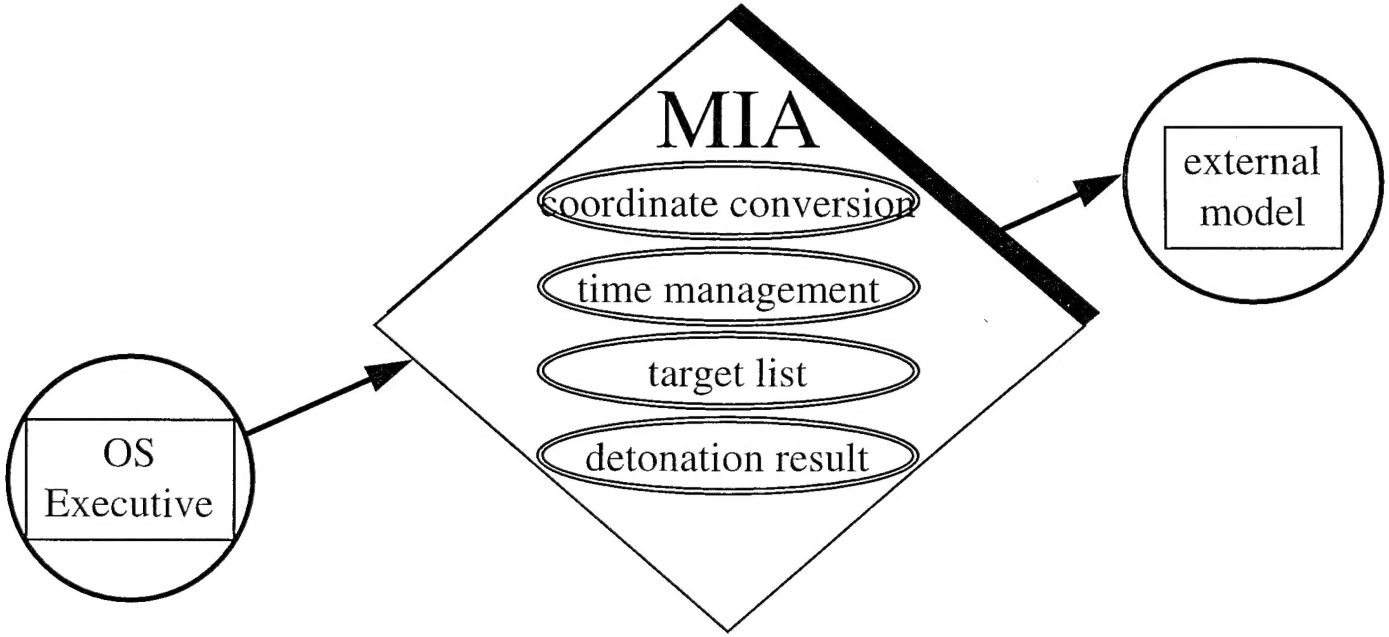
Sample OS Screen



Look Inside The Ordnance Server



Model Interface Adapter



- Make External Munition Simulation "Look Like" a Native OS Model
- Make the OS "Look Like" the Simulation Executive for the External Model

Current External Models Supported

- TACTS
  - Atoll, Aphid, Alamo, Archer
  - Sparrow, Sidewinder, Pheonix, AMRAAM
  - Stinger, Magic
  - Guideline, Goa, Ganef, Gammon, Gainful, Grail, Gecko, Gaskin, SA-16, Gobet
- Other
  - CMTGT "Tomahawk"

Ordnance Server Advantages

- Independent Munition Model Development
- Reuse of "Proven" Models
- No Additional Bandwidth
- Level Playing Field (For Munitions)
- Better Configuration Management of Munition Models

Conclusion

- Ordnance Server Offers a Reasonable Solution to Weapon Problems in DIS
- Nonintrusive Method of Adding Uniform Munition Models to Large Number Of Simulations
- Independent Munition Models Simplify Verification and Validation Process
- An Ordnance Server Has Been Demonstrated to Help Solve the "Fair Fight" Problem